C Programming

Q. What is C language? full explaination.

Ans→ C is a middle level procedural Oriented programming language developed by "Dennis Ritchie" at AT's & T bell laboratories in the year 1972 in USA.

Note :-

i) C is a compiled language, which means that programs written in C are translated into machine code before they are executed by a computer.

ii) C has a relatively small set of keywords and syntax rules, which makes it a simple language to learn and use.

iii) C is also known for its portability.

History of C language :-

⊙ Dennis Ritchie created C as an extension of B programming language, which was itself a simplified version of the BCPL programming language.

⊙ The main purpose of C was to develop the Unix operating System, which was also created at Bell Labs.

C Programming

⊙ The first version of C was developed on a DEC PDP-11 Computer running the UNIX operating System.

⊙ It was initially called "New B" but was later renamed "C".

Why C?

⊙ C programming offers several advantages :-

i) Efficiency.
ii) Portability.
iii) Flexibility.
iv) Modularity.
v) Low-level programming.
vi) Simplicity.
vii) Easy to learn.

C Programming

To run a C program :-

We need a Compiler that can translate the C code into machine code after that machine code executed by a computer.

most Commonly used C Compiler :-

- GCC
- Clang
- Microsoft visual C++
- Turbo C++

Software to run C program :-

We have many variety Softwares to run C program.

1. Dev C++.
2. VS Code.
3. Turbo C++.

Structure of c language :-

1. Documentation
   - Single line Comment
   - Multi line Comment
2. File declaration.
3. Global declaration.
4. Executable declaration.
   - function

C Programming

First C program :-

```
#include<stdio.h>
int main()
{
    printf("Learn Coding");
    return 0;
}
```

Compilation & Execution process of C++ program :-

First.C (Source Code with .C extension)
↓
Pre-processor (Links Header files)
↓
Compiler
↓
First.obj
↓
Linker (Add object files & libraries)
↓
First.exe
↓
Loader (Loader load .exe file)
↓
Program Runned

C Programming

Q. What is Variable? full explanation.

Ans→ Variable is the name of memory location where we store data, retrieved and manipulated during the program's execution.

Note :- A variable is declared with a name and a data type.

Example :- `int a = 10;`

- `int` → datatype
- `a` → variable
- `10` → stored value

Rules to declare a variable :-

i) A variable name must start with a letter (either uppercase or lowercase) or underscore.

ii) A variable name can only contain letters, digits and underscore.

iii) We can't give extra spaces between the variable.

iv) Variables are Case-Sensitive in c language.

C Programming

Types of variables :-

① Local variable :-

A variable that is declared within a function or ~~bb~~ block of code.

Note :- ⊙ It only accessible within the function or block.

Example :- ①
```
void fun()
{
    int x=10;   // local variable
}
```

②
```
if(condition)
{
    int x=10;   // local variable
}
```

C Programming

② Global variable :-

A variable that is declared outside of any function or block.

Note :- ① It can be accessed and modified from any part of the program.

Example :-

```
int x=10;

void add()
{
    printf("%d", x);
}

void sub()
{
    printf("%d", x);
}
```

C Programming

(3) Static variable :-

A variable that retains its value between function call.

Note :- (1) Static variable is only initialized once, and its value is preserved until the end of the program.

Example :-

```
void fun()
{
    static int count = 0;
    printf("%d", count);
    count++;
}

int main()
{
    fun();
    fun();
    fun();
}
```

C Programming

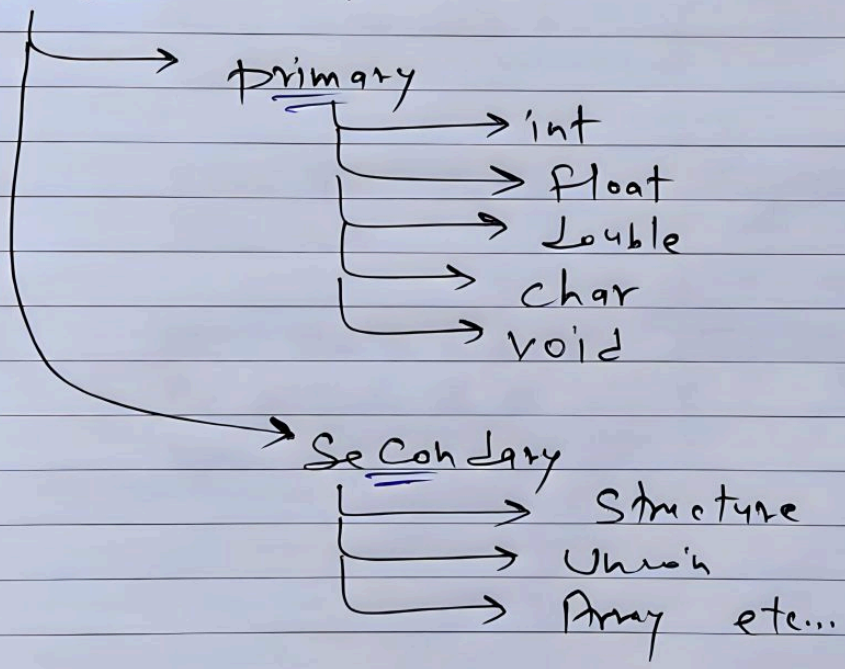
Q. What is data type? full explaination.

Ans-> Data type defines the type of value means what kind of value the variable will store.

Example:- int x = 10;

→ Here, x holding integer type of value.

Types of datatype :-



C Programming

Primary	Description	Size
int	Used for integer values.	4 bytes on most of the systems.
float	Used for floating point value with single precision.	Typically 4 bytes.
double	Used for floating point value with double precision.	Typically 8 bytes.
char	Used for single charaters.	Typically 1 byte.
void	Used ~~for~~ to represent the absense of a type.	Typically no size.

C Programming

Q. What is Constant? full explaination.

Ans→ Constant is nothing but a value that cannot be altered by the program during its execution.

Example :- Const int x=10;

→ Const is a keyword.

Note :- ① Constant can be of different types such as integer, float, character etc...

Q. What is identifier? full explaination.

Ans→ identifier refers to the name that is used to identifie variables, functions and so on.

Note :- We can't use keyword as a identifier.

Example :- i) int a=10;

→ Here, a is a identifier.

ii) int fun()

→ fun is a identifier.

{

}

iii) int int =10;

→ invalid

C Programming

Q. What is keyword? full explanation.

Ans→ Keyword is nothing but reserved word, whose meaning already defined on the Compiler.

Note:- ⊙ We can't use keyword as a variable name, function name.

Ex→ int float = 10;
↳ invalid

⊙ keyword must be in lowercase.

⊙ In C, we have total 32 keywords.

Keyword list :-

auto	extern	sizeof
break	float	static
case	for	struct
char	goto	switch
const	if	typedef
continue	int	union
default	long	unsigned
do	register	void
double	return	volatile
else	short	while
enum	signed	

C Programming

Q. What is Comment ? full explaination.

Ans → • Comments are used to add notes and explainations to the code.

• It totally ignored by the Compiler and do not affect the code execution.

Purpose to use Comment :-

i) It make code more readable.

ii) Easier to understand.

iii) And also gives detail explaination for other programmers who may read the code.

Types

Single-line Comment

// ⇒ this is single line Comment.

Multi-line Comment.

/* ..... */ this is multi-line Comment

Example:- int a=10; // a is a variable.

Example:- /* Create program for add two numbers */

C Programming

Q. What is type casting? full explaination.

Ans→ Type casting is nothing but process of converting a variable from one datatype to another.

Example :- int a=10;
float b = (Float) a;

Note :- It used when we assign a value of one datatype to a another type of variable, it can result in loss of data to avoid this problem type casting comes into the picture.

C Programming

Q. What is token? Full explaination.

Ans→ A tokens ~~is~~ are nothing but smallest individual unit of a program that is meaningful to the compiler.

Note :- ⊙ It can be a keyword, identifier, operator, constant, string, special symbol etc...

Why it important :-

Because, C compiler processes these tokens to understand the structure or syntax.

C Programming

Q. What is Operator? Full explaination.

Ans → Operator is a symbol that perfoms a specific operations on an operands (variables or values).

C language operators :-

- → Arithmetic Operator.
- → Relational operator.
- → Logical operator.
- → Assignment Operator.
- → increment / decrement operator.
- → Ternary Operator.

① Arithmetic Operator :-

Arithmetic operator performs arithmetic operations on numerical value.

Example :- ⊙ Addition (+)
⊙ Subtraction (-)
⊙ Multiplication (*)
⊙ division (/)
⊙ Modulus (%)

C Programming

(2) Relational operator :-

Relational operator compares two values and return a boolean result.

Example :-
- equal equal (==)
- not equal (!=)
- greater than (>)
- less than (<)
- greater than or equal (>=)
- less than or equal (<=)

(3) Logical operator :-

Logical operator is used to perform logical operations on boolean values.

Example :-
- Logical AND (&&)
- Logical OR (||)
- Logical NOT (!)

(4) Assignment Operator :-

Assignment operator is used to assign values to variables.

Example :-
- Assignment (=)
- Additional Assignment (+=)
- Subtraction Assignment (-=)
- Multiplication Assignment (*=)
- Division Assignment (/=)
- Modulus Assignment (%=) etc...

C Programming

(5) Increment & Decrement Operator :-

(·) Increment Operator :-

Increment operator is used to increase the value of a variable by one.

Example :-
```
int i=5;
i++;    // Now, i has a value of 6
```

Types :-
- Pre-increment (++i)
- Post-increment (i++)

(·) Decrement Operator :-

Decrement operator is used to decrease the value of a variable by one.

Example :-
```
int i=5;
i--;    // Now, i has a value of 4
```

Types :-
- Pre-decrement (--i)
- Post-decrement (i++)

(6) Ternary Operator :-

Ternary operator nothing but shorthand way of writing if-else statement. It also known as conditional operator.

Example :- `(a>b)? a:b;`

C Programming

Q. What is Control flow? Full explaination.

Ans → Control flow refers to the order in which the instructions of a programs are executed.

Control flow:-

- Conditional Statement.
- Transfer Statement.
- Iterative statement.

① Conditional statement :-

Conditional statement nothing but allow a program to execute a certain block of code only when a certain condition is true.

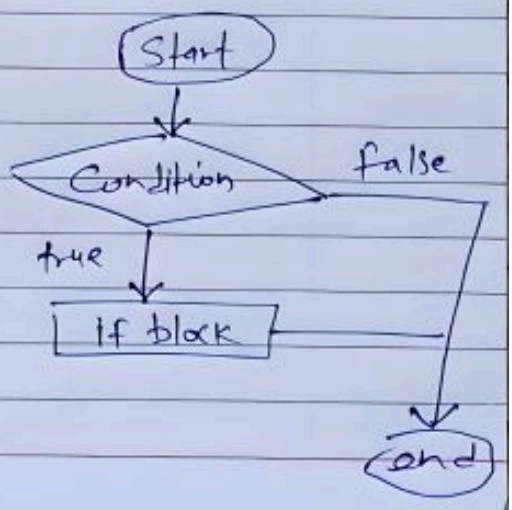
There are several types of Conditional statement :-

① if statement :- It executes a block of code only if a certain condition is true.

Syntax :-

```
if (condition)
{
    // block of code
}
```

flowchart :-



C Programming

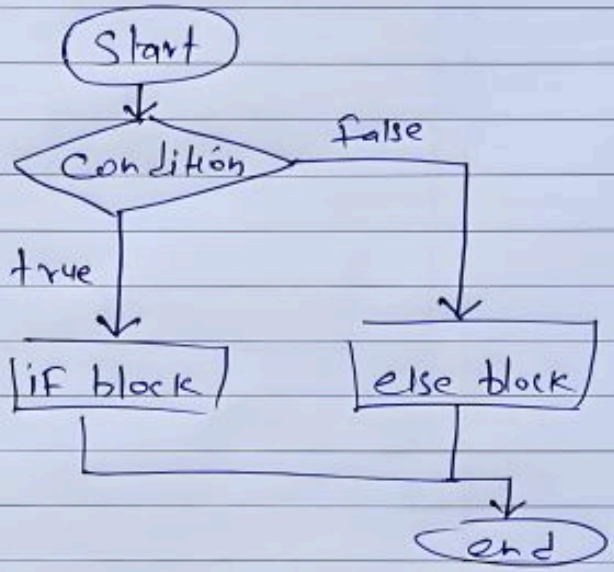
⊙ if-else Statement :-

It is used to execute two statements either if statement or else statement for a single condition.

Syntax :-

```
if( condition)
{
      // statement 1
}
else
{
      // statement 2
}
```

flowchart :-



⊙ else-if statement :-

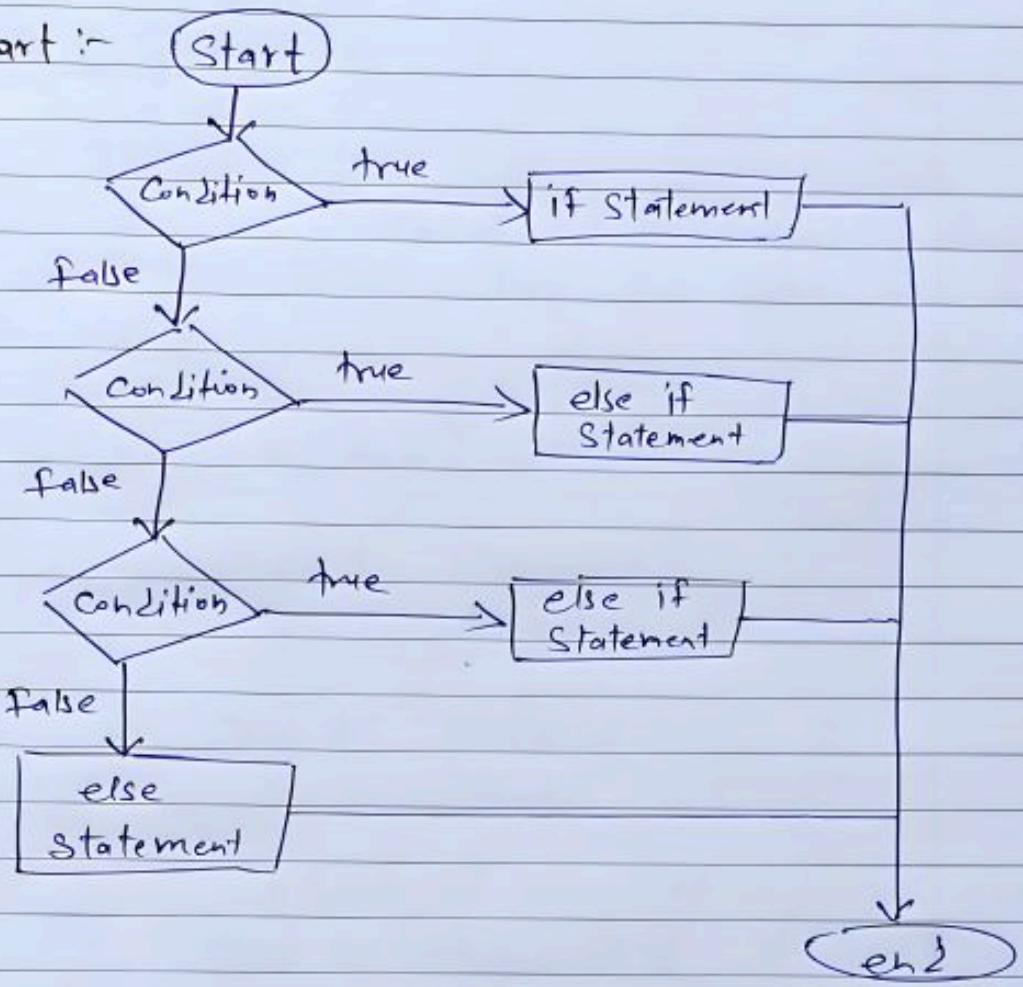
In else if we can give multiple conditions but at a time single block will be execute.

Syntax :-

```
if (condition)
{
   // code
}
else if (condition)
{
    // code
}
else
{ // code
}
```

C Programming

Flowchart :-



⊙ Nested if Statement :-

Whenever we define if statement inside this if statement we define another if block called nested if.

Syntax :-

```
if ( condition)
{
    if (condition)
    {
        // code
    }
}
```

C Programming

⊙ Switch Statement :-

Switch Statement is nothing but multiple choice Selection Statement it means when we want to Select any one case out of multiple cases.

Syntax :-

```
Switch (exp)
{
    Case 1 : Statement ;
             break ;

    Case 2 : Statement ;
             break ;
    |
    |
    |
    Case n : Statement n ;
             break ;

    Case default : Statement ;
}
```

C Programming

② Transfer Statement :-

Jumping Statement is a statement that can transfer the program control to a different part of the program.

There are four Jumping statement in C :-

(i) goto (ii) break (iii) continue (iv) return.

(i) goto :- It is used to transfer control to a labeled statement within the same function.

Syntax :- goto label;
→ it is a identifier.

Example :-
```
main()
{
    int i=1;
    label:
    printf("Learn Coding");
    i++;

    if(i<=5)
    {
        goto label;
    }
}
```

C Programming

(2) break :- Break statement is a control statement used to exit from a current iteration.

Example :-
```
for(i=1; i<=5; i++)
{
    if (i==5) {
        break;
    }
    printf("%d", i);
}
```
// output => 1 2 3 4

(3) Continue :- Continue used to skip the remaining statements inside a loop and jump to the next iteration of loop.

Example :-
```
for (i=1; i<=10; i++)
{
    if (i==5) {
        continue;
    }
    printf("%d", i);
}
```
// output =>
1 2 3 4 6 7 8 9 10

C Programming

④ return :- return statement is used to exit a function and return a value to the calling point.

Example :-

```
int add(int x, int y)
{
    return x+y;
}
main(){
    int result = add(10, 20);
    printf("%d", result);
}
```

C Programming

③ Iterative Statement :-

Iteration is nothing but loop, loop is used to repeat a block of code until the given condition is true.

C language loops :-

- while
- Do-while
- for loop

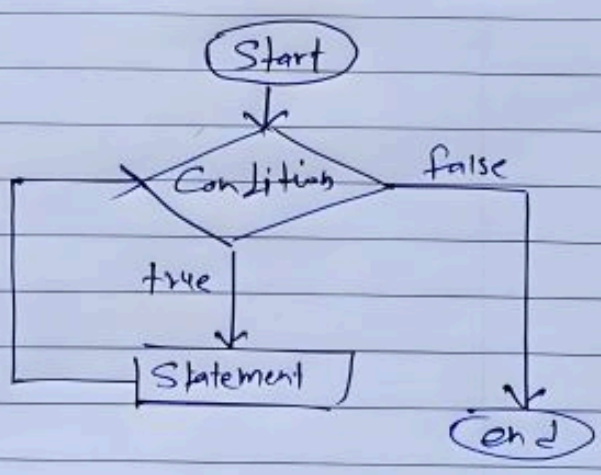
① while loop :- while loop is used when we don't know the number of iterations in advance.

Note :- It is also known as entry or pre-test loop.

Syntax :-

```
while(condition)
{
    // block of code
}
```

Flowchart :-



C Programming

⊙ Do-while loop :-

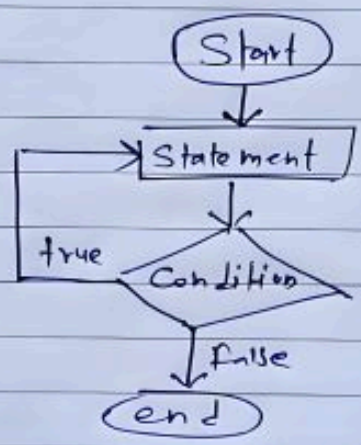
do while loop used when we want to execute loop body at least once even condition is false.

Note :- ⊙ do while loop is also known as exit or post-test loop.

Syntax :-

```
do
{
    // statement
}
while (condition);
```

flowchart :-



C Programming

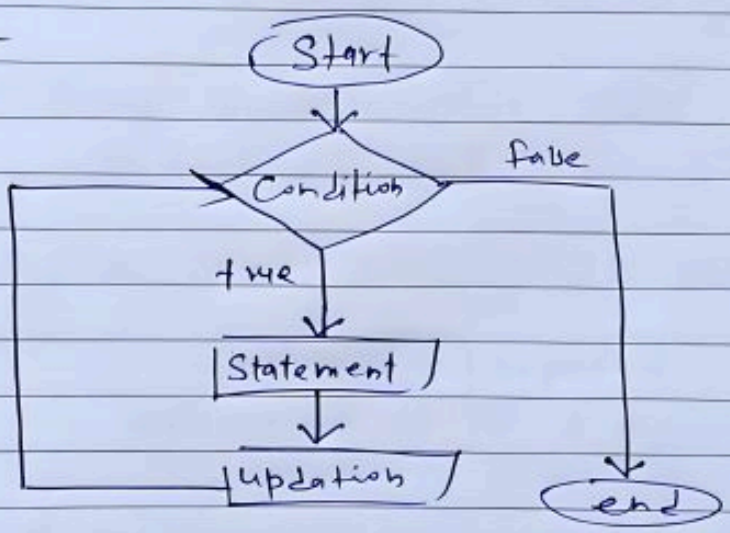
⊙ For loop :-

For loop is used when we want to perform initialization, condition and updation in single line.

Note :- ⊙ for loop is the most commonly used loop.

Syntax :-
```
for(initialization; condition; updation)
{
    // block of code
}
```

Flowchart :-



C Programming

Q. What is Array? full explaination.

Ans→ Array is nothing but collection of elements of the similar data type that are stored in contiguous memory locations.

Example:- `int arr[5] = {10, 20, 30, 40, 50};`

Note:- ⊙ Each element in array is accessed by its index.

Ex→ `int arr[5] = {10, 20, 30, 40, 50};`

`printf("%d", a[2]);` // output → 30

⊙ Array index always starts with 0 and ends with size - 1.

Ex→ `int arr[5] = {10, 20, 30, 40, 50};`

10	20	30	40	50
0	1	2	3	4

index →

⊙ Syntax:- `data_type arr_name[size];`

Example:- `int arr[5];`

Why array:- (i) Storing Collections of value in single variable.

(ii) Accessing elements.

(iii) Iterating over elements.

C Programming

Simple example of array :-

① print array elements :-

```
main()
{
    int arr[5]={10,20,30,40,50};
    printf("%d", a[2]); //output -> 30
}
```

OR

```
main()
{
    int arr[5]={10,20,30,40,50};
    for(int i=0;i<5;i++)
    {
        printf("%d", arr[i]);
    }
}
```

// output => 10 20 30 40 50

C Programming

Q. What is String? Full explaination.

Ans→ String is nothing but sequence of characters.

Example:- char str[] = "learn Coding";

Note:- String always terminated by a null character ('\0') which indicates the end of the string.

Why String :- ① By using String we can handle and manipulate textual data.

Example:- (I) take user input
(II) print string in the form of output.
(III) Copy string.
(IV) reverse string.
(V) Concatenate string.
(VI) Compare string.

② We can also manipulate string like reading input from keyboard, reading or writing files etc...

String functions in C language :-

(I) Strcpy()
(II) Strcmpi()
(III) Strcmp()
(IV) Strrev()
(V) Strcat() etc...

C language

Simple example of String :-

① print String

```
main()
{
    char str[]="Learn Coding";
    printf("%s", str);
}
```

⑪ Copy string

```
main()
{
    char str1[]="Learn Coding";
    char str2[15];
    strcpy(str2, str1);
    printf("%s", str2);
}
```

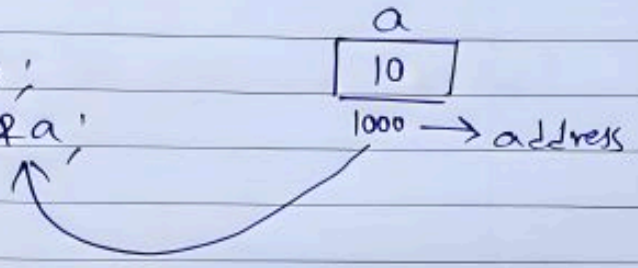
C Programming

Q. What is pointer? Full explaination.

Ans→ Pointer is a variable that holds the memory address of another variable.

Example:-

```
int a = 10;
int *ptr = &a;
```




```
printf("%d", ptr); // output => 1000

printf("%d", *ptr); // output => 10
```

Why pointer :-

① We can directly manipulate memory using pointer.

② By using pointer, we can write more efficent code and create data structure such as linked lists & trees.

Note :- (I) While working with pointer we need two unary operator.

Ex → ① & → Address of operator.
② * → Value at address operator.

(II) Syntax :- Data_type *var_name;

Example :- int *ptr;

C Programming

Note :-

(iii) Pointer variable can access the value from outside of function.

Example :-

```
void Sum(int *ptr)
{
    *ptr = *ptr + *ptr;
}
int main()
{
    int num = 10;
    Sum(&num);
    printf("%d", num);

    return 0;
}
```

C Programming

Types of pointer :-

① Void pointer

A pointer which is declared by the help of void keyword is called void pointer.

Syntax :- `void *ptr;`

Note :- ⊙ It can hold any type of address.

⊙ Also known as generic pointer.

Example :-

```
main()
{
    void *ptr;
    int a=10;
    char b='A';

    ptr = &a;
    ptr = &b;

    printf("%d", *(int *)ptr);
}
```

C Programming

(11) NULL pointer :-

NULL pointer is nothing but that does not point to any memory address or has a value of zero.

Syntax :- data_type *var_name = NULL;

Note :- It initialize with NULL value.

Example :-

```
main()
{
    int *ptr = NULL;
    printf("%d", *ptr);
}
```

C Programming

(iii) Wild pointer :-

A pointer variable that not initialized with any address. And it also known as bad pointer because of it holds the address of random memory location.

Syntax :- Data_type *Var_name;

Example :-
```
main()
{
    int *ptr;
    printf("%d", ptr);
}
```

C Programming

(IX) pointer to pointer :-

A pointer variable which holds a address of another pointer variable called pointer to pointer.

Syntax :- Data_type **var-name;

Note :- (1) It is also known as double pointer.

(2) The declaration of pointer to pointer with two asterisks (**).

Example :-

```
main()
{
    int x=100;
    int *ptr = &x;
    int **pp = &ptr;
}
```

C Programming

(v) Dangling pointer :-

A pointer variable that holds the address of in-active area location called dangling pointer.

Example :-

```
int *get_ptr() {
    int num = 10;
    int *ptr = &num;
    return ptr;
}
main()
{
    int *d_ptr = get_ptr();
    printf("%d", *d_ptr);
}
```

Note :- num is now out of scope, and d_ptr is now dangling pointer.

C Programming

(VI) Function pointer :-

A pointer variable that holds the address of function is called function pointer.

Syntax :- `data-type (*var-name)( );`

Example :-

```
int add(int x, int y)
{
    return x+y;
}
main()
{
    int (*ptr)(int, int);

    ptr = &add;

    int sum = (*ptr)(10, 20);

    printf("%d", sum);
}
```

C Programming

Q. What is Storage class? full explaination.

Ans → Storage class is nothing but scope and lifetime of a variable or function.

There are four Storage classes in c :-

- auto
- Register
- Static
- extern

Storage class	Memory	Default value	Scope	lifetime
auto	RAM	garbage	within block	till the block is active
register	register	garbage	within block	"
Static	RAM	Zero	within block	till the termination of program
extern	RAM	Zero	Anywhere in program	"

C Programming

Example :- (i) auto (ii) register (iii) static

```
void fun()
{
    auto int a=10;
    register int b=10;
    static int c=10;

    printf("%d %d %d", a, b, c);

    a++;
    b++;
    c++;

}
main()
{
    fun();
    fun();
    fun();
}
```

// output

(i) auto → 10 10 10
(ii) register → 10 10 10
(iii) static → 10 11 12

C Programming

Example:- ① extern

File1.c

```
int x=10;
```

File2.c

```
extern int x;

main()
{
    printf("%d",x);
}
```

C Programming

Q. What is function? full explaination.

Ans→ Function is nothing but block of code which takes input, processed it and produce output in the form of result.

Note :- ① Functions run only when it call.

Types :-

- User-defined.
- Pre-defined.

① User-defined function :-

A function that is created by the programmer to perform a specific task called user-defined function.

Syntax :- return-type fun-name( ) → parameters
{
// block of code
}

C Programming

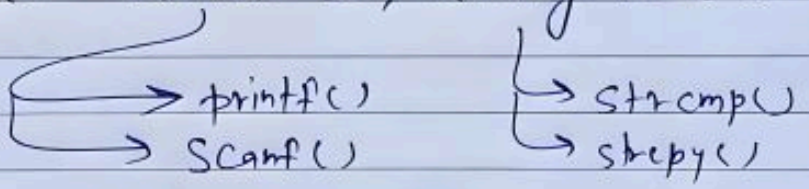
(11) Pre-defined function :-

A Functions that are built into the language and are available to use without having define them.

Note :- (1) It is also known as Standard library functions or built-in functions.

(2) They are typically included in header files.

Example :- <stdio.h>, <string.h> etc...



C Programming

Call by Value :-

In call by value the actual value of variable can't be changed, if you change the value of function parameter it is only changed for the current function.

Example :-

```
void changeValue(int a)
{
    a = a+10;
    printf("%d", a);
}
int main()
{
    int num=100;
    printf("%d", num);

    changeValue(num);

    printf("%d", num);

    return 0;
}
```

Output :-

100 → before Calling function.
110 → inside function.
100 → After Calling function

C Programming

#. Call by address :-

In call by address when a function is called by address, a pointer to the memory location of the argument is passed to the function, & any changes made to the parameter inside the function affect the orginal argument.

Example :-

```
void changValue(int *a)
{
    *a = *a + 10;
    printf("%d", *a);
}
int main()
{
    int num = 100;
    printf("%d", num);

    changeValue(&num);

    printf("%d", num);

    return 0;
}
```

Output :-

100 ⟶ before calling function.
110 ⟶ inside function.
110 ⟶ After calling function.

C Programming

Q. What is recursive function? full explaination.

Ans-> A function that calls itself called recursive function.

Note :- ① Recursive functions are used when a problem can be devided into sub-problems that are smaller instances of the same problem.

Syntax :-
```
return-type fun_name(parameters)
{
    if(base_Condition)
    {
        // code
    }
    else
    {
        fun_name(parameters);
    }
}
```

Example :-
```
int factorial(int x)
{
    if(x==0)
    {
        return 1;
    }
    else
    {
        return x * factorial(x-1);
    }
}

main(){
    int num=5;
    printf("%d", factorial(num));
}
```

C Programming

Q. What is Structure? full explaination.

Ans→ Structure is used when we want to allow different - different types of datatype into a single name.

Note:- ① It is a way to create a user-defined data type that contains related information, which can be accessed & manipulated as a single entity.

② Struct keyword is used to define structure.

Syntax:- Struct structure-name {

```
    data-type 1;
    data-type 2;
};
```

Example:- Struct emp {

```
    char name[50];
    int sal;
};
```

Note:- ① Once structure is defined, now declare variables of that type.

Ex→ Struct emp details;

declaring a variable "details" of type emp.

C Programming

Example:-

```
Struct Student {

        char name[20];
        int roll;
        float gpa;
};

int main()
{
        Struct student result;

        Strcpy( result.name, "Learn Coding");

        result.roll = 100;
        result.gpa = 4.3;

        printf("Name: %.s\n", result.name);
        printf("Roll: %d\n", result.roll);
        printf("GPA: %.2f ", result.gpa);

        return 0;
}
```

C Programming

Q. What is Union? Full explaination.

Ans→ Union is a user-defined data type that allows you to store different data types in the same memory location.

Note:- ① Union keyword is used to define union.

② Union allocates same memory for each members.

③ Size of union depends on the biggest member of union.

Syntax:-
```
union union-name {
    data-type 1;
    data-type 2;
};
```

Example:-
```
union emp {
    char name[20];
    int sal;
};
```

Note:- ① Once union is defined, now declare variables of that type.

Ex→ union emp details;

↓ declaring a variable "details" of type emp.

C Programming

Example :-

```
union Student {

    char name[20];
    int roll;
    float gpa;
};

int main()
{
    union student result;

    strcpy(result.name, "Learn Coding");

    result.roll = 100;
    result.gpa = 4.3;

    printf("Name: %s\n", result.name);
    printf("Roll: %d\n", result.roll);
    printf("CGPA: %.2f", result.gpa);

    return 0;
}
```

C Programming

Q. What is Macro? full explaination.

Ans-> Macro is a preprocessor directive that defines a name or a function-like macro that can be used throughout the code.

Note :- ① It replace the name of macro to value of macro.

② macro is defined using the '#define' preprocessor directive.

Syntax :- #define macro-name macro-value

Example :-

```
#define PI 3.14
#define Square(x) ((x)*(x))

int main()
{
    int r = 5;

    double area;

    area = PI * Square(r);

    printf("%.f", area);

    return 0;
}
```

C Programming

Q. What is file Handling? full explanation.

Ans→ file handling is nothing but storing the data in a file & accessing the data from a file which means it allow you to read from & write to files on your computer storage devices.

C allow to perform various operations on files :-

i) create a file.
ii) open file.
iii) read data.
iv) write data.
v) delete file.
vi) copy file.